The Junit Report.

Junit is a Test framework used for test-driven development for the java programming language. The objective of using a test framework is to check if the code runs the expected sequence of the function/ piece of script or fragment of code. Test driven development is a development process that is a repetition of a short development cycle. This means that the requirements of the program are turned into small very specific test cases. We then test the piece of code to check for expected results. In this report we will be describing how we used Junit testing.

For our assignment, the first test we decided to test was the list book function as one of the unit tests. The reason we did this is because our main page displays the list of books available in the repository. This is shown in the figure 1 below. The second test we decided to do was to check whether the get book information from isbn was working. The reason why we decided to test this is because when we go to update a book in the program and the isbn for the book is already set it should appear with the proper book its associated with. The unit test of the function is shown in the figure 2 below. A similar test was done for the repository function of getting the book information by its Id shown in figure 3. The purpose of this book was again for the update method which needs to get the right id associated to that book. We also created tests that are meant to fail. We did this by replacing the correct expected result to a result that was not expected in order to test the functions are correctly executing their purpose. As an example, if you are to enter an Id that does not exist, the fail id test would pass. An image of this is provided in the figure below. We also tested to check if we can delete a book by calling the delete row function of the repository. This can be seen in figure 4. The purpose of this was to check if the application is deleting the book from the database and the view of the user. This was an important one because it effects the rest of the functions as well such as list books which should not display the deleted book and update book function which should not able to update a deleted book before re adding it. Finally, we created a test case to see if you can delete all the books by calling the drop table function from the repository.

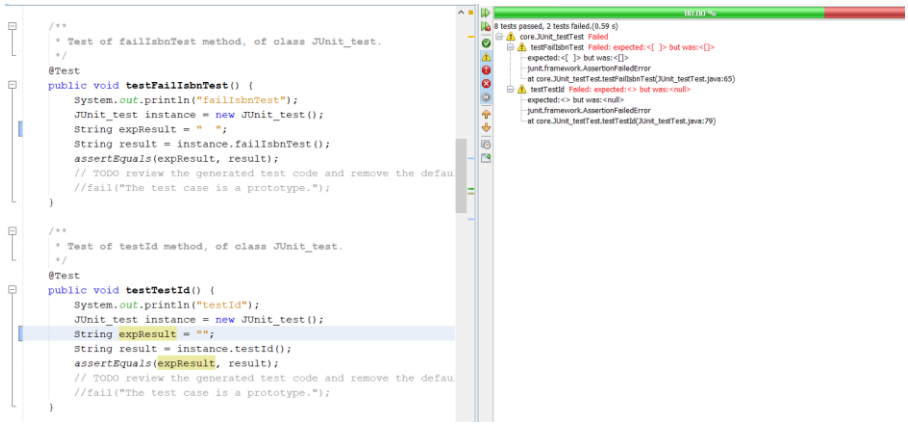
In conclusion Junit was used to help improve the program by turning requirements into very specific unit tests. For our application Junit was used to test the display of the main page, it was also used to test if the update book function was working properly by testing the infoById function and the infoByIsbn function from the repository and finally we tested the requirement of removing a book from the repository which effects the rest of the functions as well. Thus, Junit is a powerful tool used to help develop the specific requirements of a program.

```java
@Test
public void testTestBookList() {
    System.out.println("testBookList");
    JUnit_test instance = new JUnit_test();
    String expResult = instance.testBookList();
    String result = instance.testBookList();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of failTestBookList method, of class JUnit_test.
 */
@Test
public void testFailTestBookList() {
    System.out.println("failTestBookList");
    JUnit_test instance = new JUnit_test();
    String expResult = instance.failTestBookList();
    String result = instance.failTestBookList();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

(figure 1)

This figure shows how the List book Unit test is set up. The expected results are the list of books in our database. If the expected result matches the result, then the test will pass. This figure also shows a failed test where in order to pass the test must fail.



(figure 2)

This figure is the unit test to check if the Isbn will pass or not. If the expected results match the results, then the test will pass which means it is able to retrieve a book by its ISBN. The second image is a fail test where the purpose is to fail if the Isbn entered is not part of the database.

```
public void testTestId() {
    System.out.println("testId");
    JUnit_test instance = new JUnit_test();
    String expResult = instance.testId();
    String result = instance.testId();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of testFailId method, of class JUnit_test.
 */
@Test
public void testTestFailId() {
    System.out.println("testFailId");
    JUnit_test instance = new JUnit_test();
    String expResult = "";
    String result = instance.testFailId();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
   // fail("The test case is a prototype.");
}
```

(figure 3)

This figure illustrates the code to check the unit testing of the Ids.

```
@Test
public void testDeleteAllTestFail() {
    System.out.println("deleteAllTestFail");
    JUnit_test instance = new JUnit_test();
    boolean expResult = false;
    boolean result = instance.deleteAllTestFail();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of deleteRow method, of class JUnit_test.
 */
@Test
public void testDeleteRow() {
    System.out.println("deleteRow");
    JUnit_test instance = new JUnit_test();
    boolean expResult = true;
    boolean result = instance.deleteRow();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
```

(figure 4)

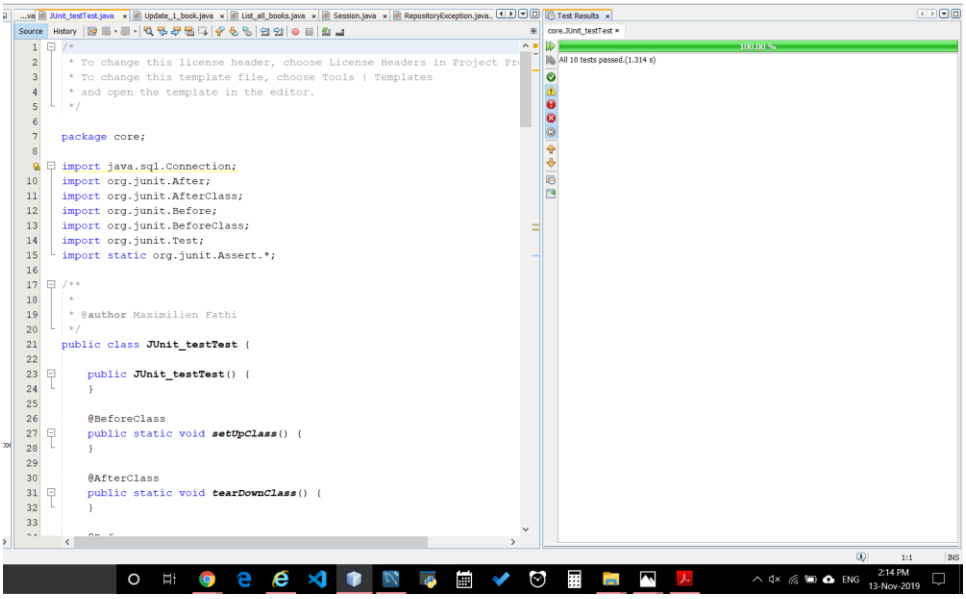
This figure shows us the test cases to check if a book can be deleted from the repository. If the book is in the repository then it should be deleted.

```
 */
@Test
public void testUpdateTest() {
    System.out.println("updateTest");
    JUnit_test instance = new JUnit_test();
    boolean expResult = true;
    boolean result = instance.updateTest();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}

/**
 * Test of updateTestFail method, of class JUnit_test.
 */
@Test
public void testUpdateTestFail() {
    System.out.println("updateTestFail");
    JUnit_test instance = new JUnit_test();
    String expResult = instance.updateTestFail();
    String result = instance.updateTestFail();
    assertEquals(expResult, result);
    // TODO review the generated test code and remove the default call to fail.
    //fail("The test case is a prototype.");
}
```

(figure 5)

This shows how the update test will work. If the book can be updated, then the test will pass by comparing the expected and results set. If they match the test will pass.



(figure 6)

This shows all the unit testing passing.